An exact approach for single machine scheduling with quadratic earliness and tardiness penalties

por

Nasim Dehghan Hardoroudi

Tese de Mestrado em Métodos Quantitativos em Economia e Gestão

Orientada por: Professor Doutor Jorge Valente

Faculdade de Economia da Universidade do Porto

2011

Biography

Nasim Dehghan Hardoroudi was born on 16th of July 1981 in Sary, Iran. She did her early schooling and unviersity preparatory studies in Qaemshahr. She then enrolled into Iran University of Science and Technology in Tehran after a highly selective admission procedure. In 2005, she finished her undergraduation studies in Mathematics with specialisation in topics of Applied Mathematics. After a brief 3-month training period spent working on updating flight managemeny system as a programmer, in 2009, she enrolled for master degree in Quentitative Methods in Economics and Management at the Faculty of Economics of University of Porto, and completed the course-work requirements in March 2011 with an average of 16.1/20.

Acknowledgments

I must thank many people for supporting me in my decision and in my efforts to pursue this Master degree. Apart from those that I mention here, the faculty of my course has been entirely understanding and supportive of my desire to pursue postgraduation in my tough times away from home. Special thanks to the director Professor Paulo Teles for the flexibility offered in following course-work.

I am deeply grateful to the huge understanding support and guidance given by my supervisor Professor Jorge Valenet from whose academic experience i have benefited immensely. His gentle patience and precious words of motivation has helped me a lot in not only finishing this period well but also in thinking further about my future academic options.

I am very thankful to my parents and my sister who have been a constant source of support, throughout my education. I am also glad to have had nice classmates and friends who made my stay smooth. And last but not the least, I am most fondly grateful to my husband Abolfazl without whose persuasion and unyielding patience this entire pursuit would not have been possible.

Abstract

In this study, we consider the single machine scheduling problem with quadratic earliness and tardiness costs, and no machine idle time. We propose two different lower bounds, as well as four lower bounding procedures. Five optimal branch-and-bound algorithms are then presented. Four algorithms incorporate the proposed lower bounding procedures, as well as an insertion-based dominance test, while the other one does not apply any lower bounding procedure.

The lower bounding procedures and the branch-and-bound algorithms are tested on a wide set of randomly generated problems. The computational results show that the branch-and-bound algorithms are capable of optimally solving, within reasonable computation times, instances with up to 20 jobs.

Resumo

Neste trabalho é considerado um problema de sequenciamento com uma única máquina, custos quadráticos de posse e de atraso e a restrição de que não é possível inserir tempo morto entre trabalhos. Dois limites inferiores são apresentados, bem como quatro procedimentos para o cálculo de um limite inferior para este problema. Cinco algoritmos baseados em partições e avaliações sucessivas são então desenvolvidos. Todos estes algoritmos utilizam um procedimento de eliminação baseado em inserções. Quatro dos algoritmos utilizam um dos procedimentos de limite inferior propostos, enquanto o quinto algoritmo não utiliza qualquer procedimento de limite inferior.

Os procedimentos de limite inferior e os algoritmos de partições e avaliações sucessivas foram testados num conjunto alargado de instâncias geradas aleatoriamente. Os resultados computacionais mostram que os algoritmos de partições e avaliações sucessivas conseguem resolver de forma óptima, dentro de tempos de computação aceitáveis, instâncias com até 20 trabalhos.

Contents

List of Tables

1 Introduction

In this paper, we consider a single machine scheduling problem with quadratic earliness and tardiness costs, and no machine idle time. Formally, the problem can be stated as follows. A set of n independent jobs $\{J_1, J_2, \cdots, J_n\}$ has to be scheduled on a single machine that can handle at most one job at a time. The machine is assumed to be continuously available from time zero onwards, and preemptions are not allowed. Job $J_j, j = 1,2, \cdots, n$, requires a processing time p_j and should ideally be completed on its due date d_j. Also, let h_j and w_j denote the earliness and tardiness penalties of job J_j, respectively. Given a schedule, the earliness of J_j is defined as $E_j = \max\{0, d_j - C_j\}$, while the tardiness of J_j is defined as $T_j = \max\{0, C_j - d_j\}$, where C_j is the completion time of J_j. The objective is then to find a schedule that minimizes the sum of the weighted quadratic earliness and tardiness costs $\sum_{j=1}^{n} \left(h_j E_j^2 + w_j T_j^2\right)$, subject to the constraint that no machine idle time is allowed.

Scheduling models with a single processor may appear to arise infrequently in practice. However, single machine scheduling environments do indeed occur in many operations (for a specific example in the chemical industry, see (Wagner *et al.* 2002). Furthermore, the performance of many production systems is quite often dictated by the quality of the schedules for a single bottleneck machine. Also, results and insights obtained for single machine problems can often be applied to more complex scheduling environments, such as flow shops or job shops.

Scheduling models with earliness and tardiness penalties are compatible with a recent trend in industry, namely the adoption of supply chain management by many organisations. In this approach, customers and suppliers try to integrate the flow of

materials, in order to improve the efficiency of the supply chain and provide a better service to the end user. The adoption of supply chain management has caused organisations to view both early and tardy deliveries as undesirable. Consequently, a significant amount of research has been done in early/tardy scheduling, as described in the survey papers (Baker and Scudder 1990), (Kanet and Sridharan 2000) (Gordon *et al.* 2002) and (Hoogeveen 2005). These are mostly concerned with the single machine scheduling where n jobs are available in advance to be processed on one machine.

Early/tardy scheduling models are also suited to the philosophy of just-in-time (JIT) production. The JIT production philosophy emphasizes producing goods only when they are needed, and therefore takes up the view that both earliness and tardiness should be discouraged. Therefore, an ideal schedule is one in which all jobs are completed exactly on their due dates. Scheduling models with both early and tardy costs are then compatible with the JIT philosophy, since jobs are indeed scheduled to finish as close as possible to their due dates.

In this paper, we consider quadratic earliness and tardiness penalties, instead of the more usual linear objective function. Therefore, deliveries that are quite early or tardy are more heavily penalized. On the one hand, as discussed in (Sun *et al.* 1999), quadratic penalties avoid schedules in which a single or only a few jobs contribute the majority of the cost, without regard to how the overall cost is distributed. On the other hand, in several practical settings, the penalties of non-conformance with the due dates do indeed increase in severity in a non-linear fashion, in line with the loss function proposed by (Taguchi 1986).

More specifically, the Taguchi loss function is equal to $L(x) = k\,(x - a)2$, where $L(x)$ is the loss to society (producer and/or customers) when one unit of a certain output is produced at level x, k is a constant and a is the ideal target level of the output. Thus, this function specifies that the loss increases quadratically as the output deviates from its desired level. In the considered scheduling problem, the jobs' due dates d_j and completion times C_j correspond to the desired target level a and the actual output level x, respectively, while the weights h_j and w_j are a generalization of the constant k. Therefore, the considered objective function is in accordance with Taguchi's loss function.

We assume that no machine idle time is allowed. This assumption is actually

appropriate for many production settings. In fact, when the capacity of the machine is limited when compared with the demand, the machine must be kept running in order to satisfy the customers' orders. Also, the assumption of no idle time is justified when the machines have high operating costs. Furthermore, idle time must also be avoided when starting a new production run involves high setup costs or times. Some specific examples of production settings where the no idle time assumption is appropriate have been given by (Korman 1994) and (Landis 1993).

This problem has been previously considered, and several heuristic approaches have been proposed. (Valente and Alves 2008) presented several dispatching heuristics, as well as simple improvement procedures, and analysed their performance on a wide range of instances. Beam search procedures were developed in (Valente 2010), while (Valente and Moreira 2009) considered several greedy randomized dispatching heuristics. Finally, genetic algorithms were developed by (Valente *et al.* 2011).

The corresponding problem with linear earliness and tardiness costs $\sum_{j=1}^{n}\left(h_j E_j + w_j T_j\right)$ has also been previously considered by several authors, and both exact and heuristic approaches have been proposed. Among the exact approaches, lower bounds and branch-and-bound algorithms were presented by (Abdul-Razaq and Potts 1988), (Li 1997), (Liaw 1999) and (Valente and Alves 2005c). Among the heuristics, several dispatching rules and beam search algorithms were presented by (Ow and Morton 1989) and (Valente and Alves 2005b, Valente and Alves 2005a), while (Li 1997) proposed a neighbourhood search heuristic procedure.

Problems with a related quadratic objective function have also been considered. The problem with a linear earliness and quadratic tardiness objective function $\sum_{j=1}^{n}\left(E_j + T_j^2\right)$ was studied by (Schaller 2004), (Valente 2007), (Valente 2008), (Valente 2009), (Valente and Goncalves 2009) and (Valente and Schaller 2010). These papers presented optimal and several heuristic approaches for both the versions with and without inserted idle time. The quadratic lateness problem $\sum_{j=1}^{n} L_j^2$, where the lateness of job J_j is defined as $L_j = C_j - d_j$ has also been studied by (Gupta and Sen 1983), (Su and Chang 1998) and (Schaller 2002), who developed both optimal and heuristic procedures. (Sen *et al.* 1995) presented a lower bound and a branch-and-bound algorithm for the weighted version $\sum_{j=1}^{n} w_j L_j^2$ where idle time is allowed only prior to the start of the first job. However, (Soroush 2009) demonstrated, via a counterexample,

that this procedure was flawed, since one of theorems used to prune nodes in the branch-and-bound algorithms was valid only for adjacent jobs, but not for non-adjacent jobs, as was assumed in (Sen *et al.* 1995). A correct branch-and-bound algorithm, along with a new dominance condition, were then presented in (Soroush 2010).

In this work, we first develop two different lower bounds. The first is based on a relaxation of the early/tardy penalties and the completion times, while the second converts the original problem to a weighted quadratic lateness problem. We then present four lower bounding procedures that incorporate these two lower bounds. Five optimal branch-and-bound algorithms that use these lower bounding procedures, as well as an insertion-based fathoming test, are also proposed. The lower bounding procedures and the branch-and-bound algorithms are then tested on a wide set of randomly generated problems.

The remainder of this paper is organized as follows. The two lower bounds and the four lower bounding procedures are described in section 2. In section 3, we discuss the implementation details of the five branch-and-bound algorithms. The computational results are presented in section 4. Finally, some concluding remarks are given in section 5.

2 Lower bounding

In this section, we propose four lower bounding procedures for the quadratic earliness and tardiness problem. We first present a lower bound based on a relaxation of both the early/tardy penalties and the completion times. Then, we propose a second relaxation that instead converts the original problem to a weighted quadratic lateness problem. An existing lower bound for the weighted quadratic lateness problem can then be used to obtain a lower bound for the original early/tardy problem. Finally, we present four lower bounding procedures.

2.1 Earliness/tardiness lower bound

In this section, we propose a lower bound based on a relaxation of the earliness/tardiness penalties and the completion times. For convenience, and without loss of generality, it will be assumed throughout this section that we wish to calculate a lower bound for a set of n jobs whose processing starts at time $t = 0$. This makes the presentation of the lower bound easier and clearer, and the extension of the lower bound to partial sequences that start at any time $t > 0$ is straightforward.

Let F denote the objective function of the quadratic early/tardy problem, i.e., $F = \sum_{j=1}^{n} \left(h_j E_j^2 + w_j T_j^2\right) = \sum_{j=1}^{n} h_j \left[\max\left(d_j - C_j, 0\right)\right]^2 + \sum_{j=1}^{n} w_j \left[\max\left(C_j - d_j, 0\right)\right]^2$. In order to derive the lower bound, we first consider a relaxation of the early/tardy penalties h_j and w_j. Let $h_{\min}$ and $w_{\min}$ respectively denote the minimum value of the earliness and tardiness penalties, i.e., $h_{\min} = \min\{h_j; j = 1, \ldots . n\}$ and $w_{\min} = \min\{w_j; j = 1, \ldots . n\}$. We now define a modified objective function Z by replacing the

earliness and tardiness penalties with $h_{\min}$ and $w_{\min}$. Therefore, the modified objective function Z is defined as $Z = \sum_{j=1}^{n} \left(h_{\min} E_j^2 + w_{\min} T_j^2\right) = h_{\min} \sum_{j=1}^{n} \left[\max(d_j - C_j, 0)\right]^2 + w_{\min} \sum_{j=1}^{n} \left[\max(C_j - d_j, 0)\right]^2$. For any given sequence, we have $Z \leq F$, since $h_{\min} \leq h_j$ and $w_{\min} \leq w_j$.

A lower bound for the quadratic early/tardy problem can then be obtained by performing a second relaxation, more specifically by relaxing the completion times C_j. Let $[j]$ denote the job in the jth position in a sequence. Also, let C_l^{LPT} be the sum of the processing times of the first l jobs, when the jobs are ordered in longest processing time (LPT) order (i.e., $p_{[j]} \geq p_{[k]}$ for $j < k$). Similarly, let C_l^{SPT} be the sum of the processing times of the first l jobs, when the jobs are ordered in shortest processing time (SPT) order (i.e., $p_{[j]} \leq p_{[k]}$ for $j < k$). We can now define an objective function Z' by replacing the completion times C_j in Z with C_l^{LPT} and C_l^{SPT}. More precisely, the modified objective function Z' is defined as:

$$Z' = h_{\min} \sum_{j=1}^{n} \left[\max(d_j - C_j^{LPT}, 0)\right]^2 + w_{\min} \sum_{j=1}^{n} \left[\max(C_j^{SPT} - d_j, 0)\right]^2.$$

For any given sequence, we have $Z' \leq Z$, since $C_j^{LPT} \geq C_j$ and $C_j^{SPT} \leq C_j$. Let d_j^{EDD} denote the due date of the jth job, when the jobs are ordered in earliest due date (EDD) order (i.e., $d_{[j]} \leq d_{[k]}$ for $j < k$). Finally, let $Z'_{EDD} = h_{\min} \sum_{j=1}^{n} \left[\max(d_j^{EDD} - C_j^{LPT}, 0)\right]^2 + w_{\min} \sum_{j=1}^{n} \left[\max(C_j^{SPT} - d_j^{EDD}, 0)\right]^2$.

Theorem 1 *$Z'_{EDD} \leq minZ \leq minF$, i.e., Z'_{EDD} is a lower bound for the optimal objective function value of the quadratic earliness and tardiness problem.*

Proof. As previously mentioned, we have $Z' \leq Z \leq F$ for any specific sequence. Therefore, it is sufficient to prove that $Z'_{EDD} = \min Z'$, since we then have $Z'_{EDD} \leq F$ for all possible sequences. This will be done by contradiction. Consider a schedule S in which jobs j and i are scheduled in positions l_1 and l_2, respectively, with $l_1 < l_2$ and $d_j \geq d_i$. Therefore, jobs i and j are not scheduled in EDD order. Also consider a schedule S' that is identical to S, except for the fact that the positions of jobs i and j have been interchanged. We must show that $Z'(S') \leq Z'(S)$.

Since only the positions of jobs i and j are different in schedules S and S', all

other jobs k, with $k \neq i, j$, occupy the same position in both schedules. The contribution of those jobs k to Z' is therefore identical in schedules S and S'. Hence, it suffices to compare the contributions of jobs i and j to Z'.

We first consider the change that may occur in the first part of Z', i.e., the change in the term $h_{\min} \sum_{j=1}^{n} \left[\max\left(d_j - C_j^{LPT}, 0\right)\right]^2$. Since $h_{\min}$ is constant, we will focus only on the expression $\sum_{j=1}^{n} \left[\max\left(d_j - C_j^{LPT}, 0\right)\right]^2$. Let $E(S)$ denote the sum of the contributions of jobs i and j to the term $\sum_{j=1}^{n} \left[\max\left(d_j - C_j^{LPT}, 0\right)\right]^2$ in schedule S. Therefore, we have $E(S) = \left[\max\left(d_j - C_{l_1}^{LPT}, 0\right)\right]^2 + \left[\max\left(d_i - C_{l_2}^{LPT}, 0\right)\right]^2$ and $E(S') = \left[\max\left(d_i - C_{l_1}^{LPT}, 0\right)\right]^2 + \left[\max\left(d_j - C_{l_2}^{LPT}, 0\right)\right]^2$. Also let $\Delta E = E(S') - E(S)$. The following three cases must then be considered.

Case 1. $C_{l_2}^{LPT} < d_i \leq d_j$. We have $E(S) = \left(d_j - C_{l_1}^{LPT}\right)^2 + \left(d_i - C_{l_2}^{LPT}\right)^2$ and $E(S') = \left(d_i - C_{l_1}^{LPT}\right)^2 + \left(d_j - C_{l_2}^{LPT}\right)^2$. Therefore, $\Delta E = \left(d_i - C_{l_1}^{LPT}\right)^2 + \left(d_j - C_{l_2}^{LPT}\right)^2 - \left(d_j - C_{l_1}^{LPT}\right)^2 - \left(d_i - C_{l_2}^{LPT}\right)^2$. Let $A = d_j - d_i$, $B = d_i - C_{l_2}^{LPT}$ and $C = C_{l_2}^{LPT} - C_{l_1}^{LPT}$. We then have $\Delta E = (B + C)^2 + (A + B)^2 - (A + B + C)^2 - B^2$, with $A, B, C \geq 0$. Expanding the squared terms and simplifying, we obtain $\Delta E = -2AC \leq 0$.

Case 2. $d_i \leq C_{l_2}^{LPT} \leq d_j$. In this case, we have $E(S) = \left(d_j - C_{l_1}^{LPT}\right)^2$ and $E(S') = \left[\max\left(d_i - C_{l_1}^{LPT}, 0\right)\right]^2 + \left(d_j - C_{l_2}^{LPT}\right)^2$. We then have $\Delta E = \left[\max\left(d_i - C_{l_1}^{LPT}, 0\right)\right]^2 + \left(d_j - C_{l_2}^{LPT}\right)^2 - \left(d_j - C_{l_1}^{LPT}\right)^2$. In order to simplify the analysis, this case can now be divided in two further subcases.

Case 2a. $C_{l_1}^{LPT} < d_i \leq C_{l_2}^{LPT} \leq d_j$. In this situation, we have $\Delta E = \left(d_i - C_{l_1}^{LPT}\right)^2 + \left(d_j - C_{l_2}^{LPT}\right)^2 - \left(d_j - C_{l_1}^{LPT}\right)^2$. Let $A = d_j - C_{l_2}^{LPT}$, $B = C_{l_2}^{LPT} - d_i$ and $C = d_i - C_{l_1}^{LPT}$. We then have $\Delta E = C^2 + A^2 - (A + B + C)^2$. Therefore, $\Delta E \leq 0$, since with $A, B, C \geq 0$ we have $(A + B + C)^2 \geq C^2 + A^2$.

Case 2b. $d_i \leq C_{l_1}^{LPT} < C_{l_2}^{LPT} \leq d_j$. We now have $\Delta E = \left(d_j - C_{l_2}^{LPT}\right)^2 - \left(d_j - C_{l_1}^{LPT}\right)^2$. Therefore, we have $\Delta E < 0$, since $d_j \geq C_{l_2}^{SPT} > C_{l_1}^{SPT}$.

Case 3. $d_i \leq d_j < C_{l_2}^{LPT}$. We have $E(S) = \left[\max\left(d_j - C_{l_1}^{LPT}, 0\right)\right]^2$ and $E(S') = \left[\max\left(d_i - C_{l_1}^{LPT}, 0\right)\right]^2$. Consequently, $\Delta E = \left[\max\left(d_i - C_{l_1}^{LPT}, 0\right)\right]^2 - \left[\max\left(d_j -\right.\right.$

$C_{l_1}^{LPT}, 0)\big]^2$. Therefore, we have $\Delta E \leq 0$, since $d_i \leq d_j$.

We now consider the change in the second part of Z', i.e., the change in the term $w_{\min} \sum_{j=1}^{n} \big[\max\big(C_j^{SPT} - d_j, 0\big)\big]^2$. Since $w_{\min}$ is constant, we will focus only on the expression $\sum_{j=1}^{n} \big[\max\big(C_j^{SPT} - d_j, 0\big)\big]^2$. Let $T(S)$ denote the sum of the contributions of jobs i and j to the term $\sum_{j=1}^{n} \big[\max\big(C_j^{SPT} - d_j, 0\big)\big]^2$ in schedule S. We then have $T(S) = \big[\max\big(C_{l_1}^{SPT} - d_j, 0\big)\big]^2 + \big[\max\big(C_{l_2}^{SPT} - d_i, 0\big)\big]^2$ and $T(S') = \big[\max\big(C_{l_1}^{SPT} - d_i, 0\big)\big]^2 + \big[\max\big(C_{l_2}^{SPT} - d_j, 0\big)\big]^2$. Also let $\Delta T = T(S') - T(S)$. The following three cases must be considered.

Case 1. $C_{l_2}^{SPT} < d_i \leq d_j$. In this case, we have $T(S') = T(S) = 0$, so $\Delta T = 0$.

Case 2. $d_i \leq C_{l_2}^{SPT} \leq d_j$. We have $T(S) = \big(C_{l_2}^{SPT} - d_i\big)^2$ and $T(S') = \big[\max\big(C_{l_1}^{SPT} - d_i, 0\big)\big]^2$. Consequently, $\Delta T = \big[\max\big(C_{l_1}^{SPT} - d_i, 0\big)\big]^2 - \big(C_{l_2}^{SPT} - d_i\big)^2$. Therefore, we have $\Delta T \leq 0$, since $C_{l_1}^{SPT} < C_{l_2}^{SPT}$ and $d_i \leq C_{l_2}^{SPT}$.

Case 3. $d_i \leq d_j < C_{l_2}^{SPT}$. In this case, we have $T(S) = \big[\max\big(C_{l_1}^{SPT} - d_j, 0\big)\big]^2 + \big(C_{l_2}^{SPT} - d_i\big)^2$ and $T(S') = \big[\max\big(C_{l_1}^{SPT} - d_i, 0\big)\big]^2 + \big(C_{l_2}^{SPT} - d_j\big)^2$. We then have $\Delta T = \big[\max\big(C_{l_1}^{SPT} - d_i, 0\big)\big]^2 + \big(C_{l_2}^{SPT} - d_j\big)^2 - \big[\max\big(C_{l_1}^{SPT} - d_j, 0\big)\big]^2 - \big(C_{l_2}^{SPT} - d_i\big)^2$. In order to simplify the analysis, this case can now be divided in three further subcases.

Case 3a. $C_{l_1}^{SPT} < d_i \leq d_j < C_{l_2}^{SPT}$. In this situation, we have $\Delta T = 0 + \big(C_{l_2}^{SPT} - d_j\big)^2 - 0 - \big(C_{l_2}^{SPT} - d_i\big)^2$. Therefore, we have $\Delta T \leq 0$, since $d_i \leq d_j < C_{l_2}^{SPT}$.

Case 3b. $d_i \leq C_{l_1}^{SPT} \leq d_j < C_{l_2}^{SPT}$. We now have $\Delta T = \big(C_{l_1}^{SPT} - d_i\big)^2 + \big(C_{l_2}^{SPT} - d_j\big)^2 - 0 - \big(C_{l_2}^{SPT} - d_i\big)^2$. Let $A = C_{l_2}^{SPT} - d_j$, $B = d_j - C_{l_1}^{SPT}$ and $C = C_{l_1}^{SPT} - d_i$. We then have $\Delta T = C^2 + A^2 - (A + B + C)^2$. Therefore, $\Delta T \leq 0$, since with $A, B, C \geq 0$ we have $(A + B + C)^2 \geq C^2 + A^2$.

Case 3c. $d_i \leq d_j < C_{l_1}^{SPT} < C_{l_2}^{SPT}$. We now have $\Delta T = \big(C_{l_1}^{SPT} - d_i\big)^2 + \big(C_{l_2}^{SPT} - d_j\big)^2 - \big(C_{l_1}^{SPT} - d_j\big)^2 - \big(C_{l_2}^{SPT} - d_i\big)^2$. Let $A = C_{l_2}^{SPT} - C_{l_1}^{SPT}$, $B = C_{l_1}^{SPT} - d_j$ and $C = d_j - d_i$. We then have $\Delta T = (B + C)^2 + (A + B)^2 - B^2 - (A + B + C)^2$, with $A, B, C \geq 0$. Expanding the squared terms and simplifying, we obtain $\Delta T = -2AC \leq 0$.

For all the situations considered, we have $\Delta E \leq 0$ and $\Delta T \leq 0$. Hence, scheduling jobs i and j in EDD order gives a lower value for $Z^{'}$. Consequently, we have $Z^{'}_{EDD} = \min Z^{'}$ and $Z^{'}_{EDD} \leq \min Z \leq \min F$, which concludes the proof.

2.2 Lateness lower bound

In this section, we propose a lower bound based on a conversion of the original early/tardy problem to a weighted quadratic lateness problem. This conversion requires a relaxation of the earliness/tardiness penalties h_j and w_j. More specifically, let $w_j^{'}$ denote the minimum of the earliness and tardiness penalties of job j, i.e., $w_j^{'} = \min\{h_j, w_j\}$. We recall that $F = \sum_{j=1}^{n} \left(h_j E_j^2 + w_j T_j^2\right) = \sum_{j=1}^{n} h_j \left[\max\left(d_j - C_j, 0\right)\right]^2 + \sum_{j=1}^{n} w_j \left[\max\left(C_j - d_j, 0\right)\right]^2$ denotes the objective function of the quadratic early/tardy problem. A modified objective function Z can now be defined by replacing the earliness/tardiness penalties h_j and w_j with $w_j^{'}$. More precisely, the modified objective function Z is defined as $Z = \sum_{j=1}^{n} \left(w_j^{'} E_j^2 + w_j^{'} T_j^2\right) = \sum_{j=1}^{n} w_j^{'} \left(E_j^2 + T_j^2\right) = \sum_{j=1}^{n} w_j^{'} L_j^2$, where $L_j = C_j - d_j$ is the lateness of job j.

The modified objective function Z corresponds to the weighted quadratic lateness problem. Moreover, for any given sequence we have $Z \leq F$, since $w_j^{'} = \min\{h_j, w_j\}$. Hence, any lower bounding procedure for a weighted quadratic lateness problem with objective function Z also provides a lower bound for the original quadratic early/tardy problem. Therefore, in order to obtain a lower bound for a given instance of the original early/tardy problem, we first create a modified quadratic lateness instance with $w_j^{'} = \min\{h_j, w_j\}$. Then, the lower bounding procedure proposed by (Soroush 2010) for the weighted quadratic lateness problem is used to calculate a lower bound for this modified instance. The lower bound presented by (Soroush 2010) improves as the sum of the processing times of previously scheduled jobs increases, since that sum is included in the lower bound's calculation. Therefore, it is to be expected that this lower bound gets better as we move further down the branching process. Actually, the procedure developed in (Soroush 2010) allows the insertion of idle time before the first job in the sequence. However, the modifications required to

adapt this procedure to a problem where idle time is not allowed are straightforward.

2.3 Lower bounding procedure

In the previous sections, we presented two lower bounds for the quadratic earliness/tardiness problem. In this section, we propose two more lower bounding procedures that incorporate these two lower bounds. In the following, let LB_ET denote the first lower bound, i.e., the lower bound based on a relaxation of the earliness/tardiness penalties and the completion times. Also, the second lower bound, based on a conversion to the weighted quadratic lateness problem, will be represented by LB_L.

We performed preliminary computational experiments with the lower bounds LB_ET and LB_L. In these experiments, we applied these lower bounds to the set of problems described in section 4 and calculated the initial lower bounds values (i.e. in these experiments all jobs are unscheduled, so the start time of the still unscheduled jobs is equal to 0). The results showed that the relative performance of the lower bounds LB_ET and LB_L was significantly influenced by the tardiness factor T. The tardiness factor of an instance (or partial sequence) is defined as $T = 1 - \left[\left(\overline{d} - t\right)/\sum p_j\right]$, where $\overline{d}$ is the average due date and t is the start time of the instance or partial sequence. When the tardiness factor is high, the average due date will be low, and most jobs will likely be tardy. Conversely, when T is low, most jobs should be completed early. Therefore, the tardiness factor T is an estimate of the proportion of tardy jobs in a schedule.

When the tardiness factor was either quite high or quite low, the two lower bounds were competitive with each other. However, lower bound LB_ET clearly outperformed the LB_L procedure for the more intermediate values of T. Indeed, for these problems, LB_ET provided a significantly higher lower bound value for a quite large number (and in some cases, actually all) of instances. We then decided to use the following lower bounding procedure, denoted by LB_ET_L_2, based on the results of these preliminary experiments. When $T < 0.1$ or $T > 0.9$, both LB_ET and LB_L are calculated, and the lower bound is then set equal to the largest of the two values. For the remaining values of the tardiness factor, only the lower bound LB_ET is used.

Furthermore, since as the time at which the unscheduled jobs increases LB_L improves, it is then possible that LB_L will become more competitive with LB_ET at the later stages of the branching process. Also, the computational tests showed that LB_L is much faster than LB_ET. We then decided to consider a lower bounding procedure in which both LB_ET and LB_L are calculated and the tighter one will be chosen as the lower bound value. This lower bounding procedure will be denoted by LB_ET_L_1.

3 Branch-and-bound procedure

In this section, we discuss the implementation details of five branch-and-bound algorithms. In fact, these five algorithms vary only in the lower bounding procedure.. Actually, we consider four branch-and-bound procedures corresponding to the four previously proposed lower bounds, as well as one algorithm in which the lower bound is simply set equal to the objective function value of the already scheduled jobs. This latter procedure is denoted by BB_NO_LB, while the four other branch-and-bound algorithms are identified by BB_LB_ET, BB_LB_L, BB_LB_L_1 and BB_LB_L_2.

The details of the branch-and-bound procedures will now be presented. We use a forward-sequencing branching rule, where a node at level l of the search tree corresponds to a sequence with l jobs fixed in the first l positions. The depth-first strategy is used to search the tree, and ties are broken by selecting the node with the smallest value of the associated partial schedule cost plus the associated lower bound for the unscheduled jobs.

The initial upper bound on the optimum schedule cost is calculated using the ETP_v2 dispatching rule, followed by the application of a 3-swap improvement procedure. This heuristic approach was the most effective of the several alternatives analysed in (Valente and Alves 2008). The upper bound value is then updated whenever a feasible schedule with a lower cost is found during the branching process.

Two fathoming tests are used to reduce the number of nodes in the search tree. In the first test, we use an insertion-based procedure to eliminate dominated nodes. This procedure inserts the job most recently added to the node's partial sequence before a certain number of the previously scheduled jobs. If an improving sequence is found, the node is then fathomed.

We considered six alternative versions of each branch-and-bound procedure.

These versions differ only in the number of previously scheduled jobs (denoted by INS) that are considered in the insertion fathoming test. In the version denoted by *single*, only one insertion is performed. In the version identified by *0.10* (respectively, *0.25*, *0.50*, *0.75* and *1.00*), on the other hand, the number of insertions that are considered is equal to 10% (respectively, 25%, 50%, 75% and 100%) of the number of previously scheduled jobs.

When the node is not eliminated by the previous test, the corresponding lower bound is then calculated for the unscheduled jobs. If the lower bound plus the cost of the associated partial schedule is larger than or equal to the current upper bound, the node is discarded.

4 Computational results

In this section, we first describe the set of test problems used in the computational experiments. Then, we analyse the performance of the four lower bounding procedures. Finally, we present computational results for the five branch-and-bound algorithms. Throughout this section, and in order to avoid excessively large tables, we will sometimes present results only for some representative cases.

4.1 Experimental design

The computational tests were performed on a set of problems with 7, 10, 12, 15, 17 and 20 jobs. These problems were randomly generated as follows. For each job J_j, an integer processing time p_j, an integer earliness penalty h_j and an integer tardiness penalty w_j were generated from one of the two uniform distributions $[45,55]$ and $[1,100]$, to create low (L) and high (H) variability, respectively. For each job J_j, an integer due date d_j is generated from the uniform distribution $[P(1-T-R/2), P(1-T+R/2)]$, where P is the sum of the processing times of all jobs, T is the tardiness factor, set at 0.0, 0.2, 0.4, 0.6, 0.8 and 1.0, and R is the range of due dates, set at 0.2, 0.4, 0.6 and 0.8.

For each combination of problem size n, processing time and penalty variability (var), T and R, 50 instances were randomly generated. Therefore, a total of 1200 instances were generated for each combination of problem size and variability. All the algorithms were coded in Visual C++ 6.0, and executed on an Intel Core2 Quad Q6600 - 2.40GHz - 3GB RAM personal computer.

4.2 Lower bound results

In this section, we analyse the performance of the four lower bounds. In Table 1, we present the average of the relative deviations from the optimum for each lower bound, calculated as $(O - LB)/O \times 100$, where O and LB represent respectively the optimum objective function value and the initial lower bound value (i.e., the lower bound at the root node). The results given in this table show that the processing time and penalty variability has a most significant effect on the performance of the lower bounding procedure. In fact, all the lower bounds perform adequately for instances with low variability, since the average deviation from the optimum is about 14%-33%. However, when the variability is high the performance is quite poor, since the lower bound value is 70% to 90% below the optimum.

These results are to be expected, given the relaxations of the earliness/tardiness penalties that were performed to obtain the lower bounds LB_ET and LB_L. In order to derive lower bound LB_ET, as previously described, the earliness and tardiness penalties of each job are replaced by their minimum values. On the other hand, the LB_L method sets the job's modified penalty to the minimum of its earliness and tardiness penalties. Therefore, the lower bound should indeed perform better when the variability of the job penalties is low. In fact, in this case the difference between the relaxed and the original penalties will be small, and the lower bound will be more accurate. When the variability of the penalties is high, however, the differences between the original and the relaxed penalty values will be larger, and the performance of the lower bound consequently deteriorates.

As the number of jobs increases, and whether the variability is high or low, the performance of the lower bounds worsens. Moreover, since we just considered the lower bounds at the root nodes, where $t = 0$, LB_L performs poorer in comparison with the other lower bounds. Naturally, the best performance is provided by LB_ET_L_1 (which always uses the best of LB_ET and LB_L), closely followed by LB_ET_L_2.

The effect of the T and R parameters on the relative deviation from the optimum for the instances with 20 jobs is given in Table 2. For the instances with a high processing time and penalty variability, the lower bounds perform better when $T = 1.0$. The relative deviation from the optimum is then similar for the remaining values of the

tardiness factor T. Also, the performance of the lower bounding procedures deteriorates as the range of due dates R increases.

When the variability is low, the lower bounds perform better when $T = 0.0$ and (especially) $T = 1.0$. Therefore, the lower bound is closer to the optimum when most jobs are early ($T = 0.0$) or tardy ($T = 1.0$). The relative deviation from the optimum then increases as the tardiness factor T approaches its intermediate values. Moreover, the performance of the lower bounds is better when the due dates are quite close ($R = 0.2$), and then deteriorates as the range of due dates R increases. Therefore, the lower bounding procedures perform worse when there is a greater balance between the number of early and tardy jobs, as well as when the due dates are more widely spread.

Furthermore, the results show that for the tardiness factor values 0.2, 0.4, 0.6 and 0.8 the performance of LB_ET, LB_ET_L_1 and LB_ET_L_2 are quite similar. In fact, when the tardiness factor T approches its itermediate values, these three lower bounds become similar. The table also demonstrates that for the instances with high variability LB_ET_L_1 and LB_ET_L_2 are competitive with each other and for the instances with low processing time and penalty variability LB_ET_L_1 is closer to the optimum value.

4.3 Branch-and-bound results

The results for the branch-and-bound algorithms will be presented in this section. In Table 3, we give the average computation times of the branch-and-bound procedures, in seconds. Due to the excessive runtimes that would be required, the BB_NO_LB and BB_LB_ET were not applied to the high variability instances with 20 jobs.

The results show that the branch-and-bound algorithms are capable of solving, within reasonable computation times, problems with up to 20 jobs. The variability of the processing times and penalties has a most significant effect on the runtimes, since the branch-and-bound procedures are much faster for the instances with a low variability. In fact, when the variability is low, it is possible to obtain an optimal solution, for the instances with 20 jobs, in about 2 seconds (on average). The average time required to solve an instance with high variability, however, is over 25 seconds.

For the instances with low variability, the computation times are somewhat close for the various INS values. The best results are given by the INS = single version, and the runtimes then increase slightly as the INS value increases. When the variability is high, however, there is a clear difference in the runtimes for the larger instances with 17 and specifically with 20 jobs. For these instance sizes, the branch-and-bound procedures with low INS values require a substantially higher computation time. Indeed, the best performance is achieved precisely for the largest values of the INS parameter (INS = 0.5, INS = 0.75 and INS = 1.00).

When the value of the INS parameter is higher, the first fathoming test considers a larger number of insertions, and the number of nodes it eliminates will then usually increase. However, the computation time required by this fathoming test also increases with the value of the INS parameter. Therefore, increasing the number of insertions has two opposite effects on the efficiency of the branch-and-bound algorithm. The results presented in Table 3 show that the outcome of this trade-off is quite different for the instances with low and high processing time and penalty variability. In fact, as mentioned above, small (large) values of the INS parameter should be used for the instances with low (high) variability.

These results are in accordance with the performance of the lower bounding procedure. Indeed, as described in the previous section, the lower bound is much closer

to the optimum for the instances with low processing time and penalty variability. Therefore, on the one hand, we would expect the branch-and-bound algorithms to be faster for instances with low variability. On the other hand, a smaller (larger) value of the INS parameter should also be necessary for instances with low (high) variability. In fact, the greater accuracy of the lower bounding procedure when the variability is low is likely to result in a larger number of nodes eliminated by the lower bound fathoming test. Consequently, it is then possible to use a smaller INS value, since the lower number of nodes fathomed by the insertion-based procedure will be offset by the higher effectiveness of the lower bound test.

Table 3 shows that, among the five branch-and-bound algorithms, BB_LB_L is the fastest. The runtimes BB_LB_ET_L_2 and BB_LB_ET_L_1 algorithms are close, even though BB_LB_ET_L_1 is somewhat faster. The worst performing algorithms are the BB_LB_ET and (particularly) the BB_NO_LB procedures. The poor performance of the BB_NO_LB procedure clearly shows that the lower bounding procedures significantly reduce the runtime of the branch-and-bound algorithms.

In Table 4, we present several additional statistics for the computation times of each branch-and-bound algorithm, namely the minimum (min) and maximum (max) values, the coefficient of variation (cov), and the percentiles 25, 50, 75, 95 and 99 (p25, p50, p75, p95 and p99, respectively) of the distribution of the runtimes. This table contains results only for the instances with 17 jobs. The results in Table 4 once again show that there is a significant difference between instances with low and high processing time and penalty variability for each branch-and-bound algorithm. When the variability is low, the best performance is obtained with the lower values of the INS parameter for a quite large number of instances in all branch-and-bound algorithms, as can be seen by the values for the percentiles 25, 50, 75, 95 and 99. Although, the INS = single algorithm is only outperformed (particularly by the INS = 0.25 and INS = 0.50) for the most difficult instances, as illustrated by the max results in BB_NO_LB and BB_LB_ET algorithms.

For instances with high variability, the procedures with a low INS value require much higher computation times. The best performance is provided by the largest values of the INS parameter. These higher INS values are also significantly more consistent, since the variability in their runtimes is lower, as indicated by the cov values.

The improvement in performance provided by the higher values of the parameter INS becomes larger as the instance difficulty increases. For the easier instances, which require low computation times, the runtimes are closer for almost all INS values, as can be seen by the min, p25 and p50 results. As the instance difficulty, and correspondingly the runtime, increase, the branch-and-bound procedures with a larger INS value become increasingly more efficient, since the increase in the runtime is much slower for these procedures. For the most difficult instances, as can be seen by the p95, p99 and max results, the computation times are substantially lower for the algorithms with a higher INS value.

As the results of Table 4 show, BB_LB_L has obtained the best performance for a quite large number of instances, specially for the most difficult ones which require higher computation times, among the other branch-and-bound algorithms, as can be seen by p50, p75, p95, p99 and max values. However, for the easier instances, BB_LB_ET_L_2 and BB_LB_ET_L_1 are competitive with BB_LB_L. As the results show, BB_NO_LB becomes faster than BB_LB_ET as the instance difficulty increases (as can be seen by the p99 and max values). Furthermore, the lower cov values of BB_NO_LB indicate that the variability in its runtimes is lower in comparison with the coefficient of variation values of the other branch-and-bound algorithms.

The effect of the T and R parameters on the computation times for instances with 20 jobs for the BB_LB_L algorithm is given in Table 5. For the instances with a low processing time and penalty variability, the branch-and-bound procedures are quite fast when $T = 0.2$, $T = 0.4$ and $T = 0.6$. The runtimes are higher for the other values of the tardiness factor, particularly when most jobs are tardy ($T \geq 0.8$). The computation times also decrease as the range of due dates R increases. When the variability is high, the procedures are quite efficient when most jobs are early (i.e., when $T = 0.0$, $T = 0.2$ and $T = 0.4$). The runtimes then increase significantly for the higher values of the tardiness factor, especially for $T = 0.6$ and $T = 0.8$.

In

Table 6, we present the average number of nodes generated by the branch-and-bound algorithm (NG), as well as the average percentage of these nodes that were eliminated by the two fathoming tests (%EL) for the five branch-and-bound algorithms, considering the instances with 17 jobs. We also provide some data on the relative importance of these tests, namely the average percentage of nodes fathomed by the lower bound test (%LB) and the insertion-based procedure (%INS). The number of nodes generated by the different branch-and-bound procedures is much larger for the instances with a high processing time and penalty variability. Also, the number of nodes decreases with the value of the INS parameter. This reduction in the number of nodes is minor for the low variability instances, but is quite significant for the problem with high processing time and penalty variability. In fact, the branch-and-bound procedures with low INS values generate a much larger number of nodes for these instances. These results are in line with the computation times previously presented in Table 3.

The proportion of nodes eliminated by the fathoming tests is somewhat higher for the instances with a low variability. Also, the percentage of eliminated nodes increases very slightly with the value of the INS parameter. The relative importance of the lower bound fathoming test is much higher for the instances with a low processing time and penalty variability. This result is to be expected, since the lower bound is much tighter for the low variability instances.

The percentage of nodes eliminated by the lower bound test decreases as the value of the INS parameter increases, while the proportion of nodes fathomed by the insertion-based dominance test correspondingly increases. Once more, this is expected, since a larger number of insertions is performed for the higher INS values.

According to the results of

Table 6 for BB_NO_LB, we can see that the average number of generated nodes is significantly higher, so with this fact the higher computation times for this branch-and-bound algorithm can be justified. Also, the low percentage of nodes eliminated by the lower bound test in this algorithm is expected, since no lower bounding procedure has been applied in this algorithm. Among the other branch-and-bound algorithms, BB_LB_ET_L_1 gives the lowest average number of generated nodes and highest proportion of nodes eliminated by the lower bound test.

In Table 7, we present the effect of the T and R parameters on the average number of nodes generated and the percentage of nodes eliminated by the lower bound test for INS = 0.75 in BB_LB_L . For instances with a low processing time and penalty variability, the number of nodes is much lower when $T = 0.2$, $T = 0.4$ and $T = 0.8$. The number of nodes generated by the branch-and-bound algorithms is then significantly higher for the remaining values of the tardiness factor, especially for $T \geq 0.8$. Also, the number of nodes decreases as the due dates become more widely spread. For instances with high variability, the number of nodes is smaller for $T \leq 0.4$. However, the number of nodes then increases substantially for the higher values of the tardiness factor when due dates become widely spread, and is particularly large for $T = 0.6$, $T = 0.8$ and $T = 1.0$. Again, these results are consistent with the runtimes previously given in Table 5. The percentage of nodes eliminated by the lower bound test is higher for instances where early jobs predominate (i.e., instances with $T \leq 0.6$). For the larger instances with high processing time and penalty variability, the insertion-based dominance procedure eliminates most of the nodes, since the percentage of nodes fathomed by the lower bound test is minor.

5 Conclusion

In this study, we considered the single machine scheduling problem with quadratic earliness and tardiness costs, and no machine idle time. Two different lower bounds were proposed. The first relaxes the early/tardy penalties and the jobs' completion times, while the second converts the original problem to a weighted quadratic lateness problem. Then four lower bounding procedures have been presented. Two of these procedures correspond with the two proposed lower bounds, while the other two use a combination of these lower bounds. We also propose five optimal branch-and-bound algorithms. Four of these algorithms incorporate the four proposed lower bounding procedures, as well as an insertion-based fathoming test. The remaining procedure does not consider any lower bounding procedure in the algorithm, though it uses the insertion-based fathoming test.

The four lower bounding procedures and the five branch-and-bound algorithms were tested on a wide set of randomly generated problems. The lower bounding procedures performed adequately for instances with low processing time and penalty variability. For high variability instances, however, the performance of the lower bounds was quite poor. The branch-and-bound algorithms were capable of solving, within reasonable computation times, problems with up to 20 jobs. Again, the performance of the branch-and-bound procedures was superior for the instances with low variability. The results show that BB_LB_L has the best performance among the branch-and-bound algorithms. Moreover, the best results were obtained by setting the INS parameter required by the insertion dominance procedure to small (large) values for the low (high) variability instances.

These results are in accordance with the relaxations of the earliness/tardiness

penalties that were performed to obtain the two lower bounds. In lower bound LB_ET, the earliness and tardiness penalties of each job are replaced by their minimum values. The LB_L method, on the other hand, sets a job's modified penalty to the minimum of its earliness and tardiness penalties. Therefore, the superior performance of the lower bounding procedure and the branch-and-bound algorithms for the low variability instances was to be expected. Indeed, for these instances there is a smaller difference between the relaxed and the original penalties, and the lower bound will be more accurate.

The proposed procedures are then quite efficient for the problems with a low penalty variability. When the variability is high, however, the performance of both the lower bound and the branch-and-bound algorithm is inferior. Nevertheless, the branch-and-bound procedures are still capable of solving high variability instances with 20 jobs in about 9 seconds.

A possibility for future research is to consider additional metaheuristics. The iterated local search (Congram *et al.* 2002), iterated greedy (Fanjul-Peyro and Ruiz 2010) and variable greedy (Framinan and Leisten 2008) metaheuristics have been shown to provide excellent results for scheduling problems, and could therefore be competitive with, or hopefully outperform, the existing genetic algorithms. Another research possibility is to extend the considered problem in order to incorporate additional elements, such as different release dates or setup times. Finally, the quadratic earliness/tardiness objective function can be used in scheduling problems with more than one machine, such as in parallel machines, flowshop and job-shop settings.

Table 1: Relative deviation from the optimum

		alg			
var	n_jobs	LB_ET	LB_L	LB_ET_L_1	LB_ET_L_2
L	7	14.270	23.436	12.181	12.985
	10	15.42	27.92	13.75	14.21
	12	16.02	29.59	14.49	14.85
	15	16.31	31.64	14.97	15.22
	17	16.46	32.32	15.23	15.39
	20	16.95	33.26	15.75	15.90
H	7	75.81	85.61	68.18	70.37
	10	81.68	89.07	75.00	76.00
	12	84.96	90.48	78.51	78.86
	15	86.94	92.27	81.49	81.62
	17	88.24	92.20	82.54	82.61
	20	89.36	93.15	84.37	84.38

Table 2: Effect of the tardiness factor and the range of due dates on the relative deviation from the optimum for the instances with 20 jobs

		low var				high var			
T	R	LB_ET	LB_L	LB_ET_L_1	LB_ET_L_2	LB_ET	LB_L	LB_ET_L_1	LB_ET_L_2
0.0	0.2	12.042	15.708	11.693	11.693	85.419	100.000	85.419	85.419
	0.4	13.960	14.910	12.667	12.667	90.921	100.000	90.921	90.921
	0.6	15.008	14.311	12.860	12.860	89.899	100.000	89.899	89.899
	0.8	15.098	13.834	12.800	12.800	90.076	100.000	90.076	90.076
0.2	0.2	14.401	26.080	14.311	14.401	86.050	100.000	86.050	86.050
	0.4	16.488	26.889	16.331	16.488	87.081	100.000	87.081	87.081
	0.6	18.395	30.244	18.012	18.395	91.143	100.000	91.143	91.143
	0.8	19.711	35.381	19.319	19.680	93.470	100.000	93.470	93.470
0.4	0.2	15.114	46.822	15.114	15.114	83.880	100.000	83.880	83.880
	0.4	19.314	62.961	19.314	19.314	88.631	100.000	88.631	88.631
	0.6	22.832	82.813	22.832	22.832	93.095	100.000	93.095	93.095
	0.8	29.448	93.331	29.448	29.448	95.627	100.000	95.627	95.627
0.6	0.2	14.863	33.487	14.863	14.863	83.993	100.000	83.993	83.993
	0.4	17.814	45.605	17.814	17.814	88.000	100.000	88.000	88.000
	0.6	20.607	68.098	20.607	20.607	89.342	100.000	89.342	89.342
	0.8	25.015	81.543	25.015	25.015	93.638	100.000	93.638	93.638
0.8	0.2	12.728	13.279	12.203	12.728	88.130	99.293	87.852	88.130
	0.4	14.926	16.304	14.243	14.926	89.365	99.892	89.365	89.365
	0.6	16.664	19.422	16.073	16.639	89.873	99.717	89.746	89.873
	0.8	17.601	21.436	16.787	17.332	92.536	99.749	92.536	92.536
1.0	0.2	12.643	7.539	7.539	7.539	83.758	43.565	43.471	43.471
	0.4	13.771	8.817	8.817	8.817	88.677	57.840	57.165	57.165
	0.6	13.817	9.311	9.296	9.296	91.395	66.150	65.996	65.996
	0.8	14.488	10.132	10.129	10.246	90.563	69.383	68.431	68.431

Table 3: Branch-and-bound runtimes (in seconds)

		Low var						High var					
		INS											
alg	n	single	0.10	0.25	0.50	0.75	1.00	single	0.10	0.25	0.50	0.75	1.00
BB_NO_LB	7	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.000	0.000	0.001	0.001	0.001
	10	0.005	0.005	0.005	0.005	0.006	0.006	0.008	0.008	0.008	0.007	0.006	0.007
	12	0.023	0.023	0.024	0.025	0.027	0.029	0.052	0.052	0.046	0.035	0.033	0.034
	15	0.218	0.219	0.230	0.246	0.266	0.281	1.316	1.336	0.745	0.450	0.392	0.407
	17	0.985	0.992	1.039	1.114	1.204	1.274	12.446	12.682	4.765	2.560	2.168	2.229
	20	9.250	9.370	9.737	10.446	11.283	11.970	.	.	.	.	.	.
BB_LB_ET	7	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.001
	10	0.002	0.002	0.002	0.002	0.002	0.002	0.007	0.007	0.006	0.006	0.006	0.006
	12	0.008	0.008	0.009	0.009	0.009	0.010	0.045	0.045	0.034	0.032	0.032	0.033
	15	0.064	0.064	0.071	0.077	0.077	0.081	1.072	1.082	0.438	0.383	0.383	0.396
	17	0.258	0.259	0.290	0.314	0.314	0.332	9.484	9.659	2.485	2.117	2.117	2.174
	20	2.328	2.352	2.576	2.774	2.774	2.943	.	.	.	.	.	.
BB_LB_L	7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	10	0.002	0.002	0.002	0.002	0.002	0.002	0.004	0.004	0.004	0.004	0.004	0.004
	12	0.006	0.006	0.006	0.006	0.007	0.007	0.019	0.019	0.019	0.016	0.016	0.017
	15	0.039	0.039	0.040	0.043	0.046	0.048	0.328	0.330	0.243	0.178	0.164	0.170
	17	0.140	0.140	0.145	0.153	0.163	0.170	2.126	2.148	1.263	0.852	0.770	0.795
	20	1.078	1.084	1.115	1.173	1.246	1.304	46.901	46.701	17.407	9.276	7.881	8.066
BB_LB_ET_L_1	7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001
	10	0.002	0.002	0.002	0.002	0.002	0.002	0.005	0.005	0.004	0.005	0.004	0.004
	12	0.007	0.007	0.007	0.007	0.008	0.007	0.023	0.023	0.022	0.019	0.018	0.019
	15	0.044	0.044	0.045	0.047	0.049	0.051	0.375	0.376	0.280	0.203	0.185	0.190
	17	0.154	0.154	0.158	0.165	0.173	0.180	2.497	2.516	1.481	0.980	0.871	0.892
	20	1.138	1.143	1.166	1.212	1.274	1.324	55.680	55.206	20.357	10.673	8.936	9.094
BB_LB_ET_L_2	7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	10	0.002	0.002	0.002	0.002	0.002	0.002	0.005	0.005	0.005	0.004	0.004	0.004
	12	0.006	0.006	0.006	0.007	0.007	0.007	0.022	0.023	0.022	0.019	0.018	0.019
	15	0.041	0.041	0.042	0.044	0.047	0.049	0.377	0.379	0.280	0.201	0.183	0.188
	17	0.146	0.146	0.150	0.157	0.165	0.172	2.522	2.540	1.484	0.975	0.866	0.888
	20	1.092	1.097	1.119	1.167	1.229	1.280	56.361	55.814	20.476	10.665	8.911	9.064

Table 4: Branch-and-bound runtimes statistics for the instances with 17 jobs

alg	var	INS	min	p25	p50	p75	p95	p99	max	COV
BB_NO_LB	L	single	0.000	0.219	1.328	1.547	1.735	2.000	2.813	68.322
		0.10	0.000	0.219	1.336	1.562	1.750	2.016	2.843	68.355
		0.25	0.000	0.235	1.406	1.640	1.781	1.993	2.515	67.402
		0.50	0.000	0.281	1.516	1.750	1.875	2.071	2.719	66.527
		0.75	0.000	0.305	1.641	1.891	2.015	2.172	2.938	65.801
		1.00	0.000	0.321	1.735	1.985	2.125	2.297	3.125	65.431
	H	single	0.001	0.040	1.801	9.766	66.068	139.206	291.667	216.410
		0.10	0.001	0.041	1.784	9.683	68.265	142.376	289.708	217.460
		0.25	0.001	0.042	1.362	5.244	22.416	44.423	68.921	181.439
		0.50	0.002	0.048	1.138	3.184	10.630	20.601	27.893	155.693
		0.75	0.002	0.056	1.095	2.854	8.228	16.057	24.296	145.398
		1.00	0.003	0.061	1.188	3.009	8.344	15.920	25.332	141.756
BB_LB_ET	L	single	0.001	0.018	0.062	0.263	1.298	1.965	2.650	169.551
		0.10	0.000	0.018	0.062	0.265	1.309	1.974	2.664	169.729
		0.25	0.001	0.021	0.071	0.301	1.449	2.108	2.878	165.308
		0.50	0.002	0.022	0.079	0.332	1.566	2.245	3.086	162.714
		0.75	0.002	0.022	0.079	0.332	1.566	2.245	3.086	162.714
		1.00	0.002	0.024	0.084	0.354	1.644	2.344	3.244	161.816
	H	single	0.002	0.044	1.003	7.742	50.076	101.128	217.050	209.662
		0.10	0.002	0.044	0.995	7.729	51.141	101.603	224.770	211.086
		0.25	0.001	0.046	0.557	3.145	9.990	20.993	30.143	160.210
		0.50	0.003	0.050	0.601	2.907	7.579	16.298	28.227	148.781
		0.75	0.003	0.050	0.601	2.907	7.579	16.298	28.227	148.781
		1.00	0.003	0.053	0.654	3.051	7.605	16.625	29.130	145.704
BB_LB_L	L	single	0.004	0.039	0.082	0.164	0.464	1.075	1.680	130.421
		0.10	0.004	0.039	0.083	0.164	0.466	1.080	1.694	130.681
		0.25	0.005	0.040	0.085	0.169	0.486	1.102	1.627	130.049
		0.50	0.005	0.042	0.089	0.180	0.510	1.154	1.694	129.599
		0.75	0.005	0.045	0.095	0.192	0.547	1.232	1.807	129.121
		1.00	0.006	0.046	0.099	0.200	0.569	1.296	1.897	129.697
	H	single	0.005	0.070	0.277	1.360	10.532	28.057	143.153	319.332
		0.10	0.005	0.070	0.277	1.364	10.577	28.688	145.367	320.629
		0.25	0.005	0.068	0.266	1.083	5.622	16.191	36.156	240.257
		0.50	0.006	0.071	0.260	0.929	3.403	8.188	17.920	187.687
		0.75	0.006	0.075	0.276	0.923	2.826	6.701	15.073	167.709
		1.00	0.006	0.079	0.288	0.962	2.928	6.704	14.391	164.111
LB_ET L	L	single	0.000	0.031	0.078	0.172	0.562	1.336	1.954	147.970
		0.10	0.000	0.031	0.078	0.172	0.563	1.344	1.968	148.796

		0.25	0.000	0.031	0.079	0.172	0.578	1.367	1.907	147.877
		0.50	0.000	0.031	0.079	0.187	0.618	1.414	1.984	147.438
		0.75	0.000	0.032	0.094	0.188	0.649	1.492	2.109	146.887
		1.00	0.000	0.032	0.094	0.203	0.672	1.562	2.188	147.896
	H	single	0.000	0.078	0.282	1.641	12.251	33.438	160.235	312.307
		0.10	0.000	0.078	0.282	1.665	12.250	33.867	162.531	313.706
		0.25	0.000	0.063	0.281	1.297	6.719	19.781	42.860	242.363
		0.50	0.000	0.063	0.274	1.118	4.031	9.774	20.453	189.969
		0.75	0.000	0.078	0.282	1.094	3.187	7.500	17.000	169.732
		1.00	0.000	0.078	0.297	1.133	3.305	7.641	16.078	166.116
BB_LB_ET_L_2	L	single	0.001	0.018	0.065	0.165	0.567	1.358	1.972	160.714
		0.10	0.001	0.018	0.065	0.165	0.568	1.363	1.976	160.913
		0.25	0.001	0.019	0.066	0.170	0.585	1.385	1.922	159.773
		0.50	0.001	0.021	0.069	0.177	0.618	1.432	2.015	158.692
		0.75	0.002	0.023	0.074	0.188	0.647	1.513	2.122	157.496
		1.00	0.002	0.024	0.077	0.194	0.674	1.575	2.205	157.646
	H	single	0.002	0.048	0.287	1.668	12.455	33.924	163.094	314.386
		0.10	0.003	0.048	0.286	1.679	12.443	34.365	165.333	315.669
		0.25	0.002	0.046	0.280	1.306	6.792	19.773	43.964	245.372
		0.50	0.003	0.048	0.274	1.083	4.000	9.874	20.654	193.259
		0.75	0.002	0.051	0.283	1.075	3.184	7.758	17.134	173.054
		1.00	0.004	0.055	0.294	1.109	3.291	7.815	16.195	169.282

Table 5: BB_LB_L runtimes (in seconds) for the instances with 20 jobs

		low var				high var			
INS	T	R = 0.2	R = 0.4	R = 0.6	R = 0.8	R = 0.2	R = 0.4	R = 0.6	R = 0.8
single	0.0	2.475	0.955	0.436	0.246	0.801	1.313	1.484	2.028
	0.2	1.658	0.565	0.281	0.173	0.197	0.159	0.192	0.554
	0.4	1.415	0.392	0.109	0.065	1.612	1.222	0.231	0.504
	0.6	1.486	0.550	0.263	0.137	115.064	44.685	23.011	10.112
	0.8	2.802	0.958	0.523	0.275	529.781	144.712	46.553	19.207
	1.0	6.962	1.846	0.866	0.437	34.183	59.771	31.130	57.109
0.1	0.0	2.482	0.957	0.437	0.247	0.804	1.317	1.489	2.026
	0.2	1.661	0.566	0.281	0.174	0.197	0.159	0.192	0.554
	0.4	1.418	0.393	0.109	0.065	1.555	1.203	0.231	0.497
	0.6	1.491	0.551	0.263	0.137	106.077	38.542	20.987	9.759
	0.8	2.824	0.962	0.525	0.276	544.260	145.262	44.949	15.725
	1.0	7.026	1.858	0.872	0.440	34.703	60.807	31.525	57.994
0.25	0.0	2.572	0.990	0.452	0.254	0.832	1.364	1.533	2.053
	0.2	1.712	0.583	0.289	0.178	0.201	0.162	0.188	0.464
	0.4	1.456	0.404	0.112	0.067	1.283	0.695	0.169	0.353
	0.6	1.537	0.569	0.272	0.141	26.541	16.265	10.933	6.015
	0.8	2.895	0.994	0.547	0.286	194.742	37.456	14.863	9.838
	1.0	7.145	1.943	0.911	0.458	30.624	29.987	15.870	15.337
0.5	0.0	2.710	1.042	0.476	0.268	0.876	1.438	1.618	2.171
	0.2	1.791	0.610	0.303	0.187	0.211	0.168	0.197	0.477
	0.4	1.525	0.423	0.118	0.070	1.350	0.678	0.169	0.290
	0.6	1.612	0.598	0.286	0.149	18.559	10.936	7.811	4.719
	0.8	3.040	1.057	0.578	0.304	79.533	20.971	9.334	7.443
	1.0	7.473	2.072	0.972	0.490	16.135	15.913	10.901	10.734
0.75	0.0	2.878	1.106	0.505	0.284	0.926	1.524	1.721	2.313
	0.2	1.897	0.644	0.321	0.197	0.221	0.177	0.209	0.510
	0.4	1.611	0.447	0.125	0.074	1.455	0.726	0.177	0.281
	0.6	1.709	0.634	0.303	0.157	16.907	10.453	6.920	4.463
	0.8	3.180	1.135	0.619	0.325	59.064	17.022	8.959	6.997
	1.0	7.955	2.227	1.047	0.526	13.855	14.340	10.235	9.683
1	0.0	2.998	1.150	0.525	0.295	0.962	1.586	1.796	2.418
	0.2	1.970	0.668	0.332	0.204	0.228	0.182	0.218	0.536
	0.4	1.672	0.464	0.129	0.077	1.539	0.780	0.189	0.294
	0.6	1.784	0.661	0.315	0.163	17.314	10.693	7.197	4.684
	0.8	3.356	1.192	0.649	0.339	58.268	17.712	9.402	7.312
	1.0	8.352	2.344	1.101	0.552	14.444	14.967	10.724	10.145

Table 6: Average number of nodes and relative importance of the fathoming tests for the instances with 17 jobs

		low var				high var			
alg	INS	NG	%EL	%LB	%INS	NG	%EL	%LB	%INS
BB_NO_LB	single	716478	88.752	12.868	87.132	8741805	85.660	16.179	83.821
	0.10	716288	88.756	12.869	87.131	8604422	85.876	15.828	84.172
	0.25	707475	88.781	12.864	87.136	3078835	86.837	15.404	84.596
	0.50	702374	88.792	12.860	87.140	1518309	87.175	15.291	84.709
	0.75	700405	88.795	12.861	87.139	1174613	87.291	15.269	84.731
	1.00	700354	88.795	12.861	87.139	1130895	87.319	15.267	84.733
BB_LB_ET	single	123651	90.489	46.777	53.223	4752623	87.343	25.446	74.554
	0.10	123640	90.491	46.775	53.225	4726167	87.427	25.037	74.963
	0.25	122124	90.510	46.741	53.259	2120221	88.072	23.281	76.719
	0.50	121067	90.516	46.713	53.287	1137876	88.298	22.616	77.384
	0.75	120430	90.519	46.711	53.289	898908	88.369	22.382	77.618
	1.00	120425	90.519	46.710	53.290	872123	88.386	22.331	77.669
BB_LB_L	single	38059	91.239	50.401	49.599	872795	89.099	31.234	68.766
	0.10	38054	91.240	50.400	49.600	867428	89.136	31.119	68.881
	0.25	37660	91.254	50.367	49.633	484370	89.616	29.110	70.890
	0.50	37382	91.260	50.337	49.663	306876	89.778	28.127	71.873
	0.75	37196	91.262	50.325	49.675	258021	89.817	27.808	72.192
	1.00	37195	91.262	50.324	49.676	253714	89.824	27.741	72.259
BB_LB_ET_L_1	single	32374	91.118	56.018	43.982	783922	89.270	33.944	66.056
	0.10	32370	91.120	56.016	43.984	781343	89.298	33.818	66.182
	0.25	32014	91.135	55.980	44.020	452893	89.726	31.661	68.339
	0.50	31765	91.141	55.950	44.050	287101	89.877	30.632	69.368
	0.75	31591	91.143	55.941	44.059	240955	89.911	30.278	69.722
	1.00	31590	91.143	55.941	44.059	236939	89.918	30.208	69.792
BB_LB_ET_L_2	single	33036	91.102	55.532	44.468	788072	89.165	33.414	66.586
	0.10	33032	91.103	55.530	44.470	785493	89.193	33.270	66.730
	0.25	32673	91.118	55.494	44.506	456584	89.633	31.059	68.941
	0.50	32420	91.124	55.465	44.535	290079	89.786	30.027	69.973
	0.75	32245	91.127	55.457	44.543	243861	89.821	29.682	70.318
	1.00	32244	91.127	55.456	44.544	239852	89.828	29.614	70.386

Table 7: Nodes generated and importance of lower bound test for INS = 0.75 in BB_LB_L

			R = 0.2		R = 0.4		R = 0.6		R = 0.8	
var	n	T	NG	%LB	NG	%LB	NG	%LB	NG	%LB
L	10	0.0	738	47.71	412	62.93	307	70.21	265	72.97
		0.2	551	54.49	316	68.31	223	75.52	181	80.16
		0.4	519	56.39	268	72.02	190	80.21	152	84.63
		0.6	658	53.04	349	67.08	245	74.62	203	78.96
		0.8	1161	40.56	505	60.04	359	66.84	298	72.52
		1.0	1628	33.33	956	47.51	499	60.40	484	61.85
	15	0.0	19930	34.02	8980	47.66	4663	57.68	3274	63.05
		0.2	14354	38.21	6413	52.06	3213	61.94	2071	69.32
		0.4	11980	41.23	4446	57.67	1690	72.71	1063	78.75
		0.6	17687	38.00	5955	55.92	2739	65.99	1942	70.58
		0.8	29138	31.78	11173	46.73	5135	58.54	3208	64.70
		1.0	64257	24.40	24085	37.72	9670	50.06	5505	57.81
	20	0.0	606647	26.25	185356	38.51	70674	49.38	35495	57.09
		0.2	370129	29.84	98804	43.85	41698	55.50	23064	61.32
		0.4	308892	32.13	62605	50.36	14407	66.40	7810	75.01
		0.6	335593	32.07	93772	48.88	38912	58.41	17872	65.65
		0.8	761178	27.41	204853	41.23	95342	50.89	43293	58.47
		1.0	2565019	15.23	504398	31.48	186861	41.58	79329	51.40
H	10	0.0	534	49.40	641	46.18	655	44.94	678	43.98
		0.2	246	63.83	259	64.73	251	65.07	367	59.25
		0.4	838	44.46	517	53.82	392	63.75	386	65.77
		0.6	2426	23.93	2103	25.81	1499	33.60	1067	41.55
		0.8	3349	25.11	2397	25.30	1968	23.96	1888	25.58
		1.0	2533	20.77	2631	20.73	2449	22.13	2128	23.90
	15	0.0	13864	31.28	14423	30.22	14256	31.18	23261	26.19
		0.2	3324	47.07	2511	52.39	3537	49.71	4318	53.55
		0.4	19046	27.96	8819	41.41	6669	53.82	3193	62.14
		0.6	122508	14.13	101980	17.01	61917	21.71	33662	28.55
		0.8	244331	16.65	88744	18.32	93646	16.21	62817	16.70
		1.0	118012	15.70	107719	17.11	104477	14.07	97717	14.28

20	0.0	197844	25.92	366014	22.62	424781	20.66	584142	20.91
	0.2	33881	40.10	29734	42.59	32063	44.96	96779	36.91
	0.4	391485	19.95	153907	32.88	28085	52.74	50006	59.44
	0.6	5108903	10.56	3245627	13.71	1963894	18.25	1170033	22.79
	0.8	20012443	15.54	5631104	13.90	2927169	12.35	2359008	11.38
	1.0	5325497	10.45	5439719	10.50	3648516	12.90	3531630	10.52

6 References

Abdul-Razaq, T.S. & Potts, C.N., 1988. Dynamic-programming state-space relaxation for single-machine scheduling. *Journal of the Operational Research Society,* 39 (2), 141-152.

Baker, K.R. & Scudder, G.D., 1990. Sequencing with earliness and tardiness penalties - a review. *Operations Research,* 38 (1), 22-36.

Congram, R.K., Potts, C.N. & Van De Velde, S.L., 2002. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *Informs Journal on Computing,* 14 (1), 52-67.

Fanjul-Peyro, L. & Ruiz, R., 2010. Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research,* 207 (1), 55-69.

Framinan, J.M. & Leisten, R., 2008. Total tardiness minimization in permutation flow shops: A simple approach based on a variable greedy algorithm. *International Journal of Production Research,* 46 (22), 6479-6498.

Gordon, V., Proth, J.M. & Chu, C.B., 2002. A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research,* 139 (1), 1-25.

Gupta, S.K. & Sen, T., 1983. Minimizing a quadratic function of job lateness on a single-machine. *Engineering Costs and Production Economics,* 7 (3), 187-194.

Hoogeveen, H., 2005. Multicriteria scheduling. *European Journal of Operational Research,* 167 (3), 592-623.

Kanet, J.J. & Sridharan, V., 2000. Scheduling with inserted idle time: Problem taxonomy and literature review. *Operations Research,* 48 (1), 99-110.

Korman, K., 1994. A pressing matter. *Video.* 46-50.

Landis, K., 1993. Group technology and cellular manufacturing in the westvaco los angeles vh department, *Project Report in IOM* 581.

Li, G., 1997. Single machine earliness and tardiness scheduling. *European Journal of Operational Research,* 96 (3), 546-558.

Liaw, C.F., 1999. A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem. *Computers & Operations Research,* 26 (7), 679-693.

Ow, P.S. & Morton, T.E., 1989. The single-machine early tardy problem. *Management Science,* 35 (2), 177-191.

Schaller, J., 2002. Minimizing the sum of squares lateness on a single machine. *European Journal of Operational Research,* 143 (1), 64-79.

Schaller, J., 2004. Single machine scheduling with early and quadratic tardy penalties. *Computers & Industrial Engineering,* 46 (3), 511-532.

Sen, T., Dileepan, P. & Lind, M.R., 1995. Minimizing a weighted quadratic function of job lateness in the single machine system. *International Journal of Production Economics,* 42 (3), 237-243.

Soroush, H.M., 2009. A note on "minimizing a weighted quadratic function of job lateness in the single machine system". *International Journal of Production Economics,* 121 (1), 296-297.

Soroush, H.M., 2010. Single-machine scheduling with inserted idle time to minimise a weighted quadratic function of job lateness. *European Journal of Industrial Engineering,* 4 (2), 131-166.

Su, L.H. & Chang, P.C., 1998. A heuristic to minimize a quadratic function of job lateness on a single machine. *International Journal of Production Economics,* 55 (2), 169-175.

Sun, X.Q., Noble, J.S. & Klein, C.M., 1999. Single-machine scheduling with sequence dependent setup to minimize total weighted squared tardiness. *IIE Transactions,* 31 (2), 113-124.

Taguchi, G., 1986. *Introduction to quality engineering : Designing quality into products and processes*: Tokio, Japan : Asian Productivity Organization.

Valente, J.M.S., 2007. Heuristics for the single machine scheduling problem with early and quadratic tardy penalties. *European Journal of Industrial Engineering,* 1 (4), 431-448.

Valente, J.M.S., 2008. An exact approach for the single machine scheduling problem with linear early and quadratic tardy penalties. *Asia-Pacific Journal of Operational Research,* 25 (2), 169-186.

Valente, J.M.S., 2009. Beam search heuristics for the single machine scheduling problem with linear earliness and quadratic tardiness costs. *Asia-Pacific Journal of Operational Research,* 26 (3), 319-339.

Valente, J.M.S., 2010. Beam search heuristics for quadratic earliness and tardiness scheduling. *Journal of the Operational Research Society,* 61 (4), 620-631.

Valente, J.M.S. & Alves, R.A.F.S., 2005a. Filtered and recovering beam search algorithms for the early/tardy scheduling problem with no idle time. *Computers & Industrial Engineering,* 48 (2), 363-375.

Valente, J.M.S. & Alves, R.A.F.S., 2005b. Improved heuristics for the early/tardy scheduling problem with no idle time. *Computers & Operations Research,* 32 (3), 557-569.

Valente, J.M.S. & Alves, R.A.F.S., 2005c. Improved lower bounds for the early/tardy scheduling problem with no idle time. *Journal of the Operational Research Society,* 56 (5), 604-612.

Valente, J.M.S. & Alves, R.A.F.S., 2008. Heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties. *Computers & Operations Research,* 35 (11), 3696-3713.

Valente, J.M.S. & Goncalves, J.F., 2009. A genetic algorithm approach for the single machine scheduling problem with linear earliness and quadratic tardiness penalties. *Computers & Operations Research,* 36 (10), 2707-2715.

Valente, J.M.S. & Moreira, M.R.A., 2009. Greedy randomised dispatching heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties. *International Journal of Advanced Manufacturing Technology,* 44 (9-10), 995-1009.

Valente, J.M.S., Moreira, M.R.A., Singh, A. & Alves, R.A.F.S., 2011. Genetic algorithms

for single machine scheduling with quadratic earliness and tardiness costs. *The International Journal of Advanced Manufacturing Technology,* 54 (1), 251-265.

Valente, J.M.S. & Schaller, J.E., 2010. Improved heuristics for the single machine scheduling problem with linear early and quadratic tardy penalties. *European Journal of Industrial Engineering,* 4 (1), 99-129.

Wagner, B.J., Davis, D.J. & Kher, H.V., 2002. The production of several items in a single facility with linearly changing demand rates. *Decision Sciences,* 33 (3), 317-346.